In The Claims:

Claim 1 (Currently Amended): A method for mapping a valid stack up to a destination program counter, comprising:

mapping a path of control flow on the stack from any start point in a selected method to the destination program counter by locating a linear path from the beginning of the method to the destination program counter and iteratively processing an existing bytecode sequence for each branch, and identifying said path as complete when said destination program counter is reached; and

upon finding a path, simulating stack actions for executing said existing bytecodes along said path, and constructing a virtual stack for storage in a pre-allocated memory location;

Claim 2 (Original): The method of claim 1 wherein the step of mapping a path of control flow on the stack comprises:

processing a first linear bytecode sequence until the control flow is interrupted; and recording unprocessed targets from any branches in the first linear bytecode sequence for future processing.

Claim 3 (Original): The method of claim 2 wherein the step of mapping a path of control flow on the stack further comprises:

processing an additional bytecode linear sequence until the control flow is interrupted; and

recording unprocessed targets from any branches in the additional linear

bytecode sequence for future processing, where the destination program counter was not reached during an earlier processing of a linear bytecode sequence.

Claim 4 (Original): The method of claim 2 wherein the step of processing any linear bytecode sequence comprises:

determining if a bytecode in said any linear bytecode sequence is a breakpoint with a pointer to bytecode data; and

replacing the breakpoint with the bytecode data.

Claim 5 (Original): The method of claim 3 wherein the step of processing any linear bytecode sequence comprises:

determining if a bytecode in said any linear bytecode sequence is a breakpoint with a pointer to bytecode data; and

replacing the breakpoint with the bytecode data.

Claim 6 (Original): The method of claim 1 wherein the step of simulating stack actions executing the bytecodes along the path further comprises generating a virtual stack.

Claim 7 (Original): The method of claim 6, further comprising:

encoding the virtual stack as a bitstring and storing the bitstring at a selected destination for use in memory management operations.

Claim 8 (Original): The method of claim 7, wherein the step of storing the bitstring comprises

storing the bitstring to the selected method as compiled on a heap.

Claim 9 (Original): The method of claim 7, wherein the step of storing the bitstring comprises storing the bitstring to a pre-allocated area on the stack.

Claim 10 (Original): The method of claim 1 wherein the step of simulating stack actions executing the bytecodes along the path further comprises:

inserting pre-determined stack actions for bytecodes maintaining the control flow in the selected method; and

calculating stack actions for bytecodes transferring the control flow from the selected method.

Claim 11 (Currently Amended): A method for mapping a Java bytecode stack up to a destination program counter comprising:

mapping a path of control flow on the stack from any start point in a selected method to the destination program counter by locating a linear path from the beginning of the method to the destination program counter and iteratively processing an existing bytecode sequence at each branch, and identifying said path as complete when said destination counter is reached; and

upon finding a path, simulating stack actions for executing said existing bytecodes along said path, and constructing a virtual stack for storage in a pre-allocated memory location.

Claim 12 (Original): The method of claim 11 wherein the step of mapping a path of control flow on the stack comprises:

processing a first linear bytecode sequence until the control flow is interrupted; and

recording unprocessed targets from any branches in the first linear bytecode sequence for future processing.

Claim 13 (Original): The method of claim 12 wherein the step of mapping a path of control flow on the stack further comprises:

processing an additional bytecode linear sequence until the control flow is interrupted; and

recording unprocessed targets from any branches in the additional linear bytecode sequence for future processing, where the destination program counter was not reached during an earlier processing of a linear bytecode sequence.

Claim 14 (Original): The method of claim 12 wherein the step of processing any linear bytecode sequence comprises:

determining if a bytecode in said any linear bytecode sequence is a breakpoint with a pointer to bytecode data; and

replacing the breakpoint with the bytecode data.

Claim 15 (Original): The method of claim 13 wherein the step of processing any linear bytecode sequence comprises:

determining if a bytecode in said any linear bytecode sequence is a breakpoint with a pointer to bytecode data; and

replacing the breakpoint with the bytecode data.

Claim 16 (Original): The method of claim 11 wherein the step of simulating stack actions executing the bytecodes along the path further comprises generating a virtual stack.

Claim 17 (Original): The method of claim 16 further comprising:

encoding the virtual stack as a bitstring and storing the bitstring at a selected destination for use in memory management operations.

Claim 18 (Original): The method of claim 17, wherein the step of storing the bitstring comprises storing the bitstring to the selected method as compiled on a heap.

Claim 19 (Original): The method of claim 17, wherein the step of storing the bitstring comprises storing the bitstring to a pre-allocated area on the stack.

Claim 20 (Original): The method of claim 11 wherein the step of simulating stack actions executing the bytecodes along the path further comprises:

inserting pre-determined stack actions for bytecodes maintaining the control flow in the selected method; and

calculating stack actions for bytecodes transferring the control flow from the selected method.

Claim 21 (Original): A computer-readable memory for storing the instructions for use in the execution in a computer of the method of claim 1.

Claim 22 (Original): A computer readable memory for storing the instructions for use in the execution in a computer of the method of claim 11.

Claim 23 (Currently Amended): A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for mapping a valid stack up to a destination program counter, said method steps comprising:

mapping a path of control flow on the stack from any start point in a selected method to the destination program counter and identifying said path as complete when said destination counter is reached; and

upon finding a path, simulating stack actions for executing existing bytecodes along said path, wherein the step of mapping a path of control flow on the stack comprises:

processing a first linear existing bytecode sequence until the control flow is interrupted; and

recording unprocessed targets in a pre-allocated memory location from any branches in the first linear existing bytecode sequence for future processing, and

where the destination program counter was not reached during an earlier processing of a linear existing bytecode sequence,

processing an additional existing bytecode linear sequence until the control flow is interrupted; and

recording unprocessed targets in said pre-allocated memory location from any branches in the additional linear existing bytecode sequence for future processing.